4/24/2023

Actor-Critic Algorithm

- Actor-critic Algorithm – separate V/Q value calculation and policy learning. The difference is that both get updated simultaneously.
- **Sidenote:** Expanding TD equation to infinity converges to Monte-Carlo equations
- Actor-critic algorithm utilizes the preference metric (from MAB problem – softmax/gradient policy)

$$\pi(a_t|s_t) = \frac{e^{H(s_t,a_t)}}{\sum_{a\in A} e^{H(s_t,a)}}$$

$$H(a_t) = H(a_t) + \alpha \frac{\partial E[R]}{\partial H_a}$$

$$H(a) = H(a) + \alpha[R_a - \bar{R}][1'_{aa} - P_{a'}]$$

- In summary, two equations that are updated every step. Actor, which is the policy, and then the critic, which is the V or Q value.
    - o Critic:

$$V(s_t) = V(s_t) + \alpha.\delta_t$$

        - where δ_t is the TD-0 error: $\delta_t = [R_{t+1} + \gamma V(s_{t+1}) - V(s_t))]$
    - o Actor:

$$H(s_t, a_t) = H(s_t, a_t) + \beta.\delta.\left(1 - \pi(a_t|s_t)\right)$$

- From Prof. Mahdi's HW3

Tabuler Actor-Critic Algorithm

$V(s)=0$, $H(s,a)=0$, for all $s \in S$, $a \in A$.

Repeat for N episodes

- Start from a random state $s_0 \in S$, $t=0$

  While $t<T$ (episode Length).

  - Select action: $a_t \sim \pi(\cdot|s_t)$: $\pi(a|s) = \frac{e^{H(s,a)}}{\sum_{a\in A} e^{H(s,a)}}$
  - Take action a_t, move to state s_{t+1} and observe R_{t+1}.
  - $\delta_t = R_{t+1} + \gamma V(s_{t+1}) - V(s_t)$
  - $V(s_t) = V(s_t) + \alpha \delta_t$
  - $H(s_t,a_t) = H(s_t,a_t) + \beta \delta_t (1 - \pi(a_t|s_t))$
  - $t = t+1$

- Very fast convergence due to the gradient being used directly compared to other algorithms (policy + value iteration, SARSA + Q-learning, etc.)
- Also model-free, similar to SARSA + Q-learning
- Any policy can be used (doesn’t have to be softmax). The only difference would have to be that the gradient would have to be computed accordingly, leading to a difference in the H or preference equation.

Summary

- MDP - <S, A, R, P(s’|s, a)>
- If MDP is known, model-based policies
  - Policy iteration
  - Value iteration
- If MDP is not known, model-free policies
  - Monte-Carlo
    - Generate a bunch of trajectories, estimate V values based on G_ts
  - Temporal Difference learning
    - Instead of going depth-wise, choose single action (or more, depending on TD length)
      - SARSA
      - Q-learning
  - Actor-Critic
    - Similar to TD, but separation between policy and value function
- All these algorithms were tabularized, we had discrete tables. A real problem might not be tabularized. Actions, states, can be continuous (for example, car position/speed in Cartpole). Tabular algorithms no longer work ~ would converge too slowly/consume too much memory.
- **One solution:** approximate and discretize the action/state spaces. Tradeoff between closeness to true state space and time to converge.

Continuous RL Algorithms

- **Option 1:** Instead of learning Q(s, a), learn theta (set of constants?) i.e. Q\theta(s, a, \theta)
- **Option 2:** DNN – states and action as input, Q value output. Another network can be used for the policy. Typically, 2-3 layers are enough to get good estimates. Question: How is the network trained?
- **Option 3:** Wavelets/Fourier transforms
- **Option 4:** Simple polynomials
  - $Q(s,a) = w_1 s.a + w_2 s^2.a + w_3 s.a^2$
  - $Q(s,a) = [w_1, w_2, w_3].[\phi_1(s,a), \phi_2(s,a), \phi_3(s,a)]^T$
  - In this case, we’re using known basis functions (phi) to express our Q-values as polynomials
  - **Example:**
    - $V^{\pi}(s) \sim \tilde{V}(s,w) = w^T.\phi(s)$
    - If phis are chosen as indicator function i.e. $1(s = s_1) = 1\ if\ s = s_1, 0\ otherwise$
      - $\tilde{V}(s,w) = w^T.\phi(s) = [w_1, w_2, \dots w_n].[1(s_1), 1(s_2), \dots 1(s_n)]$ – this is the tabular case
    - Any class of phis can be chosen

Least Square Policy Iteration (LSPI)

- $\pi^0 \rightarrow PE \rightarrow V(s) \rightarrow PI \rightarrow \pi^1 \rightarrow \cdots$
- Policy Evaluation PE:

$$V_{k+1}(s) = \sum_{s'} P\big(s' \big| s, \pi(s)\big) \left[R + \gamma V_k(s')\right]$$

Or

$$Q^{\pi}(s,a) = \sum_{s'} P(s'|s,a) \left[R + \gamma Q(s', \pi(s'))\right]$$

- Policy Improvement

$$\pi'(s) = argmax_{a \in A}\, Q(s,a)$$

For model-free approach:

$$Q^{\pi} = R + \gamma . MQ^{\pi}$$

Solving for Q,

$$Q^{\pi} = (I - \gamma M)^{-1} . R$$

If Q is large, not possible to compute inverse in reasonable amount of time

So we apply the approximation principles as follows:

$$Q(s,a;w) = \sum_{j} \phi_j(s,a). w_j$$

See paper: https://users.cs.duke.edu/~parr/jmlr03.pdf

- Works well when state space is continuous,, but not as much when action nspace is continuous